Preproduction Performance of the SGI Origins on the NAS Workload

Robert J. Bergeron

Report RND-xxx January 2000

Abstract

This paper reports floating-point and related performance measurements for SGI Origin machines executing the NAS workload over a 1-year preproduction period. Both machines, the 64-CPU *hopper* and the 256-CPU *steger,* employed R10000 processors and displayed about 50% user CPU utilization. The SGI Event Counters reported that the average performance rate per *hopper* CPU exceeded 17 MFLOPS. Refinement of this measurement led to a best estimate of 28 MFLOPS per CPU, with the typical user job utilizing about 10 CPUs and averaging about 0.5 GFLOPS. For *steger,* the counters reported that the average rate per CPU exceeded 15 MFLOPS. Refinement of this measurement led to a best estimate of 24 MFLOPS per CPU with the typical user job utilizing about 18 CPUs and averaging about 0.8 GFLOPS. NAS production codes execute on a 16-CPU C90 with a 1 GW memory, and relative to user jobs executing on this platform, the average jobs on *hopper* and *steger* displayed performance improvement factors of 1.2 and 2.0, respectively. A simple analysis indicates that most jobs in the NAS Origin workload are memory-bound on the R10k processor. Since memory speed determines the floating-point performance of the NAS workload, processors claiming to increase the current performance of the average NAS user job will need to increase memory speed as well as improve CPU performance. However, these SGI Origins are the first RISC machines which provide, for the typical NAS user, a measured level of performance superior to that of the C90.

1. Introduction

In science, the first principle is that you must not fool yourself, and you're the easiest person to fool.
-Richard Feynman

In high performance computing, each new chip and architecture oversells its virtues and understates its drawbacks. Reports describe how an architecture composed of these chips produces "hundred-GFLOP" performance and imply that such rates represent user workload performance. Computer sites even provide measured system performance of machines that have never executed a user workload. Such exaggeration overwhelms the reality of high performance computing - the performance experienced by the typical user.

This report presents the workload-measured floating-point performance of two NAS SGI Origin machines, the 64-CPU *hopper* machine and the 128/256-CPU *steger* machine. On a selected user application and under special operating conditions, *steger* can display a floating-point performance exceeding 20 GFLOPS. However, the major goal of this paper is to establish how these machines perform for the typical user executing a typical application under normal operating conditions. The workload reports generated by various system utilities executing on the Origins will provide the data required to establish the performance level.

The NAS workload is a general purpose, computationally-intense, batch job mix. Users demand system resources in nonoptimal and unpredictable order, and every day the demand and the order changes. No

single code dominates system performance; the workload consists mostly of Computational Fluid Dynamics (CFD) programs including grid generators, aerodynamic flow solvers, and multidisciplinary design optimization codes. The flow solvers employ block-tridiagonal and multigrid-relaxation algorithms for steady-state problems and utilize FFT techniques in direct numerical simulations of turbulence. This highly vectorized workload is floating-point and memory intensive, displaying a sustained rate near 4 GFLOPS on a 1993 vintage Cray C90 parallel vector supercomputer containing 16 CPUs with 1 GW of memory [1]. The performance limiting factor for this workload was the vector length of 78, which was 60% of the length required to sustain peak performance. Hardware measurements indicated that the instruction issue delay due to memory operations was a small fraction of the time required for a workload memory reference. The workload requires about 1 floating-point memory operation for each floating-point operation (FLOP) on the C90, and about 97% of all memory references are floating-point. Measured I/O rates to SSD and rotating disks for this C90 total about 32 mB/sec, of which the disk I/O is about 5 mB/sec. The bulk of NAS C90 I/O is directed to the SSD as auxiliary memory and temporary storage for executing programs. The high-speed memory requirement for the average NAS C-90 job about 70 megawords (MW).

This report uses floating-point operations (FLOPS) as the measure of machine performance. The canonical measure of parallel performance is time-to completion, but workload monitoring does not permit such a measure. Floating point operation rate continues to be a good measure of performance because parallelism for most of the NAS codes involves a single program multiple data(SPMD), domain decomposition with one or more processors per domain. There is little evidence of users changing solution techniques from implicit to explicit algorithms to avoid global communication while artificially increasing the FLOP rate. Low FLOP rate, sparse matrix techniques, which employ logical operations to reduce floating-point operations, do not appear to contribute significantly to NAS workload operation count.

The measurements reported here occurred before these machines entered production status. In preproduction periods, because the system is available to a limited number of users, the fraction of time the system is busy with user applications may be less than production periods as administrators repair "wear-in" hardware and software problems. Key utilities may not be available; for example, the lack of a checkpoint utility may require administrators to drain batch queues to execute large multi-CPU jobs. NAS uses the Portable Batch System (PBS) [2] scheduler to allocate system resources and during the preproduction period, the system administrators may modify this scheduler to, for example, impose new queue and queue limits for improved resource allocation. Compilers and other software utilities may also change frequently to reflect bug fixes. To encourage user participation, accounting charges for user jobs may be reduced or dropped altogether, and this period may see a considerable amount of user code development with perhaps shortened execution times.

The measurements provided in this report include the IRIX System Activity Reports (SAR), the accounting records maintained by PBS, and hardware monitor data reported by the SGI Event Counters. Section 2 describes the SGI Origin Architecture and NAS administrative setup. Section 3 discusses the System Activity Reports to quantify the level of user activity in the preproduction period. Section 4 analyzes the accounting results to obtain the number of CPUs requested and utilized by NAS users. Section 5 reports the Event Counter results and discusses the reasons for the current level of performance. Section 6 uses the SAR utility, accounting records, and Event Counter measurements to place the results in perspective. Section 7 presents some concluding remarks.

2. SGI Origin Architecture and NAS Administration

The Origin Architecture consists of pairs of MIPS R10000 processors sharing a common memory and connected into nodes by hardware components termed hubs. Hardware components termed routers connect groups of nodes. The following discussion provides some of the key features of the Origin architecture [3,4] with its R10000 (R10K) processor and describes how NAS manages its Origin cluster.

The R10K is a 250 mHZ superscalar RISC processor, capable of fetching and decoding 4 instructions per cycle. These instructions can execute on 5 independent pipelined functional units: a load/store unit, 2 arithmetic logic units, a floating-point add unit, and a floating-point multiply unit. The latter two units can be chained together to perform multiply-add instructions. The peak performance for this chip is 2 floating-point operations per clock cycle, but the chip can achieve this rate only if it operates on "in-cache" data. The R10K provides a two-level cache hierarchy: a 32 kB 2-way set associative Level 1 on-chip cache with a 64 byte line size and a 4 mB 2-way set associative Level 2 off-chip secondary cache with a 128-byte line size. The time to access data is 2-3 clock cycles for data in the L1 cache and 8-10 clock cycles for data in the L2 cache. Both caches are non-blocking and the L2 cache is interleaved to increase memory bandwidth. If the processor must access data in user main memory, i.e., the data is not already in the primary or secondary cache, the latency is at least 60 clock cycles. The Origin employs a virtual memory Operating System (OS) which arranges user main memory into areas called pages. For each executing process, the CPU has access to a directory, termed the Translation Lookaside Buffer (TLB), containing a partial list of the addresses of pages currently in memory. If the program references a page not in the TLB, a TLB miss occurs as the OS must update the directory to reference the required page. While different delays are associated with the various levels of TLB misses, the typical TLB delay experienced is about 70 cycles per miss. The R10K chip includes these enhanced cache and memory features to provide strong floating-point performance.

The basic Origin building block is the node, a system consisting of two R10K processors with a hub chip to manage processor access to both the common memory and to the I/O port(s). For systems larger than two nodes, the Origin employs router hardware to control the information flow between the nodes. A single router has 6-ports, allowing it to connect to up to 6 other hubs or routers. The Origin uses a high-speed network, termed the Craylink interconnect, to move data between local and remote memory which may be located on different nodes.Larger number of processors assume a hypercube topology to minimize remote memory access delays. The network hardware enables globally-addressable programming of the Origin Distributed Shared Memory (DSM), but access to this physically distributed memory is no longer uniform in time. Users exploit the multiprocessor Origin by using either message-passing, typically with MPI [5], or shared-memory programming, most recently with OpenMP [6]. Users employing shared-memory programming will observe that access to the memory in nodes closer to the executing processor requires less time than access to nodes further away. This **N**on-**U**niformity of **M**emory **A**ccess (NUMA) introduces potential scaling problems into parallel codes which access arrays distributed across the Origin nodes. This shared memory arrangement also introduces timing reproducibility problems, both in MPI and in shared memory applications, because the batch system may, at one time assign a nearest-neighbor arrangement of nodes to execute the job, and at another time, a non nearest-neighbor configuration. While SGI has spent a considerable amount of effort in creating a cache-coherent DSM, the preproduction NAS workload ran MPI applications almost exclusively and so the current paper will emphasize absolute performance as opposed to scalability and mechanisms which enforce scalability.

This report describes the performance of two machines, *hopper* and *steger*, which comprise the compute server part of the NAS Origin cluster. *Hopper* was a 64-CPU Origin machine throughout the monitoring period. For the first 126 days of the monitoring, *steger* was a 128-CPU machine, but improvements in SGI interconnect technology allowed *steger* to be combined with another 128-CPU machine and the combined 256-processor system was monitored for the remaining 259 days. Each pair of CPUs, termed a node, shared 512 mB of local memory.

During the preproduction period, a multiprocessor Origin named *turing* acted as the cluster's front end, handling logins, compiling, and small debugging jobs. A single scheduler controlled both of the cluster's batch machines. Scheduling was node-based with user charges based on the fewest number of nodes required to meet both memory and processor requirements. Scheduling priority on the Origin cluster varied to fit user needs. During prime time, shorter jobs took priority with jobs running no longer than two hours of wall-clock time. Longer jobs had priority during non-prime time as jobs could run for up to eight hours.

For the NAS systems, differences of 20% in elapsed time for the same code in workload execution were common, prompting NAS administrators in the preproduction period to configure the system into 2-CPU building blocks to provide reproducible job timings. The number of CPUs a user obtained for his job would always be a multiple of the 2-CPU building block, and while this arrangement promoted reproducibility, it also increased the number of idle processors. This increase is over and above the number of idle processors expected due to the scheduling problem in parallel systems. For example, the SP2 PBS scheduler was unable to sustain utilization exceeding 90% because turnaround requirements demanded a cache of idle processors to provide rapid turnaround for short jobs [7].

3. SAR

IRIX updates a number of software counters, termed the System Activity Report (SAR), whenever a significant event occurs in the operating system. The set of counters related to performance measurement reports the mode: user, system (including interrupt and I/O) or idle, in which the CPU finds itself when the event occurs. Several other system commands report statistics describing CPU activity and a question may arise over the most appropriate tool for reporting CPU utilization. Perhaps the PBS *qstat* command or the IRIX command *uptime* also would report legitimate results.

The PBS *qstat* report of CPU utilization will exceed the SAR CPU usage because:

- Users base their CPU request upon either the memory or the CPUs required by their job. A single-processor job requiring 4*512 mB would require 8 CPUs. During the execution of that job, 7 CPUs would be idle.
- The minimum number of CPUs PBS assigns to a job is two, so PBS reports no single processor jobs. This minimum commitment is an Origin-specific mod; PBS on the IBM SP class of machine does not necessarily have this minimum. The PBS minimum commitment, as employed on the Origin, avoids memory contention and subsequent performance problems which can arise if two different jobs execute on the same node and share memory. However, 64 single-CPU jobs submitted continuously to the 64-CPU *hopper* machine via PBS would produce an actual user CPU utilization of only 50% because PBS reserves 2 CPUs for each single-CPU job.

The IRIX command *uptime* provides moving averages of the number of processes in the run queue. This number includes both processes connected to the CPU and those trying to execute. Thus, the number of running processes reported by *uptime* will be greater than or equal to the number processes actually executing. U*ptime* will report a CPU utilization exceeding 100% for excessively loaded machines. Any conversion of *uptime* reports into an average CPU utilization should allow no more than the maximum number of CPUs to contribute to this result. Table 1 compares the results of the three techniques on *hopper* and *steger* for the preproduction period.

Table 1: Comparison of NAS Origin System CPU Utilization (%) Reports

Measurement	*hopper*	*steger*
SAR	53.6	47.3
uptime	56.7	47.8
qstat -r	80.8	76.7

The *qstat* utility assumes that all CPUs assigned by PBS are executing user jobs and, as discussed above, this assumption leads to overestimates of CPU utilization. Use of the *uptime* utility to represent CPU usage assumes that all processes in the run queue connect to CPUs and experience with *uptime* has shown

that this assumption overestimates the number of running processes.
The SAR reports that the CPU utilization (user plus system) for the Origin platforms is about 50%. Current NAS experience with these machines indicates that a CPU utilization approaching 100% is not an achievable goal. Initial workload testing indicated that the machines tend to experience performance and stability problems when employing all CPUs to execute user workloads and a 90% CPU utilization may be the maximum CPU activity the Origins can sustain.
Figures 1 and 2 present the SAR-based system utilization for the 64-CPU *hopper* and the 128/256-CPU *steger*. *Hopper* utilization averaged 54% with a standard deviation of 17%-the usage increased apparently until the *steger* machine became available. To promote *steger* usage, NAS administrators designated it as a "nocharge" machine-users could run their jobs with no decrease in their CPU allocations. *Steger* utilization averaged 47% with a standard deviation of 20%. Figure 2 shows that the average utilization for 128-CPU and 256-versions of *steger* are quite similar.

4. Accounting Records

The PBS Server daemon provides a job-based accounting log which contains the number of CPUs requested, elapsed time, CPU time, and job timestamps. PBS logs thus allow the calculation of an effective number of CPUs, defined as the ratio of CPU time to elapsed time. For example, if the highly parallel NPB LU code requests 32 CPUs, the PBS accounting records will report that the ratio of CPU time to elapsed time was 31.05. This ratio indicates that apart from some small amount of single-CPU pre- and postprocessing, LU utilized 32 CPUs. However, the number of jobs executing on the system and the number of CPUs used by these jobs are correlated [8]. The number of CPUs utilized by the average *hopper* and *steger* job cannot be found by any sort of averaging of accounting file aggregates. Instead, continuous real-time monitoring and a simple averaging of the results must be performed to give the appropriate result. Although such monitoring was not done during the preproduction period, the accounting records provide enough information to simulate this monitoring. Ordering the jobs in the PBS accounting files in time and marching forward from the first job in 600-second intervals simulates a continuous monitoring of these systems. Counting jobs and both their requested and utilized CPUs gives the results shown in Table 2.

Table 2: Number of CPUs for the typical Origin job

Machine	CPUs	standard deviation
hopper requested	13.6	11.1
hopper utilized	10.2	9.1
steger requested	23.3	21.0
steger utilized	18.3	13.8

The large standard deviations shown for both machines are expected because the scheduler will try to execute large jobs and backfill with smaller jobs, so typically the mix of jobs running on these batch machines will include a mix of both many-CPU jobs and few-CPU jobs. The average number of CPUs utilized takes on added importance in the computation of the average user job performance-only the CPUs actually utilized by the job contribute to values reported by the Event Counters.

Figures 3 and 4 display the distribution of CPUs requested by the user jobs, weighted by the PBS elapsed time for each job and then normalized by the total elapsed time of all jobs for that machine. This approach attempts to give some estimate of the true system demand, independent of the processor scheduling constraints. The *hopper* distribution is quite symmetric about the central maximum of 32, half the number of machine CPUs. User jobs requesting less than 32 CPUs are much more numerous than those at the

higher end. On *steger*, the time-averaged peak also occurs at 32 CPUs, indicating a less-than-expected user demand to apply additional CPUs to their jobs.

The PBS accounting records also allow identification of the users responsible for the majority of CPU usage. Review of the user source files and user job execution times indicate that the vast majority of the NAS Origin codes use the MPI programming model and that over 97% of the user execution time involved the MPI over the Craylink interconnect. Few of the NAS users currently exploit Origin's globally-addressable distributed memory via loop-level parallelism or the MPI shared memory (shmem) utility. Based on NAS SP-2 workload measurements [15], the number of CPUs requested and utilized on *steger*, as shown in Table 2, is fewer than expected. Users employing the MPI programming model may be finding it difficult to scale their code to utilize the large number of processors available on the Origins.

The accounting records also supply the queue wait times, defined as the time interval between job submission and the time the job begins execution. The SGI Origin machines have many processors and one might expect a many-processor supercomputer to service user jobs faster than a few-processor supercomputer. The NAS few-processor supercomputer is a 16-CPU Cray C90, *von neumann (vn)*, which also employs PBS to service batch jobs. The PBS service split between debug and regular queues on the Origins occurs at two hours elapsed time. PBS will service requests for any number of CPUs with elapsed time less than two hours as soon as possible. But batch service for the *vn* queues is primarily based upon CPU time restrictions and the *vn* debug queue is 300 CPU seconds. While it is difficult to provide an equivalent service division for the two architectures, Table 3 summarizes the Origin and C90 queue wait delays based upon a two-hour elapsed time limit for the Origins and a two-hour CPU time limit (including debug jobs) for the C90.

Table 3: Average Queue Wait Times (Minutes)

Service	Cray *vn*	Origin *hopper*	Origin *steger*
less than 2-hour request	48	45	97
more than 2-hour request	257	648	270

It is surprising that SGI Origin machines with 50% idle display relatively long queue wait times, but jobs requesting more than half of the available CPUs experience most of this delay. During the preproduction period, the Origin administrators did not have the checkpoint utility to suspend user jobs. Any large jobs, i.e., jobs requesting more half the number of CPUs, required the administrators to drain the queues and this activity added to the wait time and machine idle.

5. Event Counters

This section describes the measurement of system floating-point performance and presents some discussion drawn from these measurements regarding the observed level of performance. The R10K processor supplies two counters, termed Event Counters [9], for reporting certain hardware events. Each of these counters can track one event at a time and provides a choice of 16 events per counter. The events report the type and number of instruction executed, including memory and floating-point, as well as the type and number of cache misses. Event Counters do not distinguish between the types of memory operations corresponding to floating-point, integer, and address operations, nor do they provide direct information on memory bank contention.

The counters support two different modes of operation: global and local. Global mode reserves the counters for root, allowing performance measurement of the entire system. Local mode allows the users to monitor performance of their individual processes, almost always their executable programs. NAS

uses the PBS scheduler to mediate between the two modes. The NAS SGI systems default to global mode and switch to user mode when a batch job notifies the PBS scheduler that it wishes to use the Event Counters. About 70% of all CPU cycles were monitored during the preproduction period with this back-and-forth approach.

Values accumulated by the Event Counters, for example floating-point instructions, are reported typically on an hourly basis and the rates are based on the elapsed time. Workload performance can be measured in terms of system performance and/or CPU performance and, since this report uses both of these measures, the following text clarifies their meaning. If the 64-CPU *hopper* Event Counters report that

64*30.e6*3600

floating-point instructions were executed in the previous hour, then *hopper* system performs a rate of 30 floating-point MIPS/CPU during that hour. If the SAR indicates that the fraction of time spent in user and system mode totalled 50%, then the *hopper* CPUs (or codes running on *hopper* CPUs) were performing at 60 floating-point MIPS/CPU during that hour.
The hardware counters on the NAS Origins do not distinguish between floating-point multiply, add, and multiply-add instructions. Since a multiply-add instruction generates two floating-point operations, floating-point MIPS are typically less than floating-point MFLOPS. For 22 benchmark codes, comparison of the number of floating-point operations reported by the Cray Hardware Performance Monitor and by the SGI Event Counters indicated that the SGI floating-point operations were underreported, on average, by a factor of 1.6. In this report, floating-point rates reported by the SGI Event Counters will include a factor of 1.6 to provide an estimated floating-point rate. In the preceding example, the estimated *hopper* CPU floating-point operation rate would be 96 MFLOPS/CPU.

Table 4 shows the floating-point performance rates sustained for these two machines in the preproduction period.

Table 4: Estimated Floating-Point Performance (GFLOPS) for the SGI Origins

Machine	Estimated Performance	Standard Deviation
hopper	1.78*	0.80
steger(128/256)	4.08/6.18*	1.49/2.34

*- Result includes a factor of 1.6 to compensate for underreporting floating-point operations

The rates correspond to 28 MFLOPS/CPU for *hopper* and 24 MFLOPS for the 256-CPU *steger*. Evidently, doubling of the *steger* processors did not produce a doubling of performance in the preproduction period. This result is interesting because the system utilization was about the same for the 128 and 256 periods. Moreover, the machine does scale very well on applications using large number of processors. [The average number of processors used by steger in its 128-CPU incarnation was 17 as opposed to a usage of 18 in the 256-CPU period.

Figure 5 and 6 present the floating-point performance reported for the two machines. On *hopper*, the system performance increases slowly and then slowly decreases, displaying the same behavior as the system utilization. Figure 5 shows a performance spike to about 40% of peak at day 575. Examination of the accounting logs revealed that the user application was a multiprocessor semiconductor device modelling code employing MPI and Level 3 BLAS to solve a linear system of equations. Although this code used a considerable amount of CPU time on day 575, the code was still executing test problems. The much larger real cases must be carefully coded to ensure local memory access in order to display such high performance[10].

Since these measurements treat actual multiprocessor workloads, their complexity and realism provides a limited ability to isolate performance effects.Under these circumstances, interpreting performance results with a hardware counter involves comparing the counter output of several high-performing codes with the application or workload in question. Raw counts, such as the number of cache misses, must be placed into a context, such as the ratio of cache misses to memory operations, to generate a meaningful measurement. The Event Counters provide some context-based statistics, which shed more light on program performance than the raw counts themselves, so the discussion below will focus on these statistics. Table 5 provides an explanation for several of the non-obvious statistics.

Table 5: Definitions of Key Event Counter Statistics

Statistic	Comment
Primary cache line reuse	Number of times, on average, that a primary data cache line is used after it has been moved into the cache.
Secondary Cache Line Reuse	Number of times, on average, that a secondary data cache line is used after it has been moved into the cache.
Primary Data Cache Hit Rate	Fraction of memory accesses satisfied from a cache line already resident in the primary data cache; computed as: 1 - ratio of L1 cache misses to the sum of memory loads and stores.
Secondary Data Cache Hit Rate	Fraction of memory accesses satisfied from a cache line already resident in the secondary data cache; computed as: 1 - ratio of L2 cache misses to L1 cache misses
Time accessing memory/Total time	Sum of the estimated times of loads, stores, primary data cache misses, secondary data cache misses, and TLB misses, divided by the total program run time.
Primary-to-secondary bandwidth used	Rate of data movement between the primary and secondary data caches (megaBytes per second).
Memory bandwidth used	Rate of data movement between the secondary data cache and main memory (megaBytes per second).

Table 6 provides a comparison of the SGI Event Statistics for a high-performing code with those of a measured workload interval. The code is a RISC-optimized CFD program, ARC3D[11]. The measured workload interval, labelled WRK, represents one of the higher performing intervals, for which the SAR reported 100% user utilization.

Table 6: Comparison of SGI Event Counter Statistics

Event Statistic	ARC3D	WRK	Comment
Instructions/cycle	0.96693	1.11338	
Floating-point instructions/cycle	0.40268	0.15632	ARC3D more efficient
Loads & stores/cycle	0.34944	0.46392	
Loads & stores/floating-point instruction	0.86777	2.96775	ARC3D data reuse
L1 Cache Line Reuse	18.6215	18.5680	
L2 Cache Line Reuse	7.02746	29.3948	
L1 Data Cache Hit Rate	0.94904	0.94889	
L2 Data Cache Hit Rate	0.87543	0.96710	
Time accessing memory/Total time	0.81854	0.74649	
L1--L2 bandwidth (mB/s)	208.859	276.688	
Memory bandwidth (mB/s)	99.1731	36.8000	
MFLOPS (average per process)	105.671	39.0800	ARC3D-20% of peak

The ARC3D code employed double precision variables and, since more single precision variables can be stored in a single cache line than double precision variables, comparison of cache hit rates can be a bit misleading unless both programs use the same type of variables. This table illustrates three general points:

- To generate floating-point operations, a high-performance code engages in far less memory activity (loads and stores) than typical NAS workload codes. Relative to WRK, ARC3D displays a greater L2 cache-to-memory bandwidth transfer rate and the higher ARC3D L2 cache miss rate produces this result.
- A high performance code requires fewer auxiliary instructions, such as address calculations, to generate floating-point operations.
- A high-performance code does not necessarily display high data cache hit rates.

If the ARC3D code is a typical representative of R10K-optimized code, this comparison indicates that optimization of NAS codes requires a global approach to reduce memory references.
The ratio of FLOPS to memory references for WRK is 0.336, much lower than the 1.0 ratio historically characteristic of the C90 NAS workload[1]. The reason for the C90's ability to generate floating-point operations with fewer memory references is the large amount of local storage which was provided precisely to avoid memory loads and stores. In addition to 8 64-element floating-point vector registers, the C90 provides 8 address registers and 8 scalar registers, each having a set of 64 backup registers. The SGI Origin has 64 single-element hardware registers, which must serve as floating-point, integer, and address registers.

A simple analysis[12] shows how memory limits the R10K performance. The average time to access memory, *tavg*, is the sum of the probability-weighted access times for each of the three memory hierarchy

components: L1 cache, L2 cache, and main memory. The frequency of memory access, *fma*, is the average number of memory instructions per cycle. When *tavg* exceeds *fma* clock cycles, memory speed determines system performance because the time to service a memory access exceeds the time to execute the instructions intervening between two memory accesses.

In the ARC3D case, Table 6 shows that the frequency of memory access, *fma*, occurs every 2.862 cycles (1/0.34944).
The average time to access memory is

tavg = p1 * t1 + (1 - p1) * p2 + (1 - p1) * (1 - p2) * tm, where

p1(2) = probability of a L1(2) cache hit
t1(2) = L1(2) cache access time
tm = memory access time

With lower-bound access times from Section 2 and Table 6,

tavg = 0.949 * 2 + (1 - 0.949) * 0.875 * 8 + (1 - 0.949) * (1 - 0.875) * 60.

The use of tm = 60 cycles implies that all accesses to memory were local, i.e., to the memory connected to the CPU, and not over the DSM network. The result, *tavg* = 2.294 cycles, is less than *fma*, so the code is not memory-bound. However, if t1=3 cycles and t2=10 cycles, *tavg* = 3.657 cycles. Since *tavg* exceeds *fma*, the ARC3D R10K performance would be memory-bound. This approach does not allow a clear verdict on whether the code is memory-bound or not.

In the WRK case, for t1=2 cycles and t2=8 cycles, *tavg* = 2.394 and *fma*= 1/0.46392 = 2.156. The calculations performed in this 1-hour workload interval were memory-bound because, under the minimum hardware memory access times, the average memory access time exceeded the frequency of memory access. Since WRK represents one of the better-performing periods, many of the NAS workload codes appear to be memory-bound on this system. Over 33% of the workload intervals displayed memory-bound performance and only 5% of the monitored intervals displayed performance which was not bound by memory.

The Event Counter performance results reflect the strong dependence of NAS workload codes on memory operations and the achievable memory bandwidth associated with the R10k processor. An often-quoted example is the STREAM benchmark[12], which shows that the memory bandwidth of the Origin system ranges from 358 mbyte/sec for a single R10K 250 mHZ processor to 192 mbyte/sec for 64 R10K processors. This rate translates to 90 MW/sec for single precision data and 45 MW/sec for double precision data. If the NAS codes required about 1 memory operation per floating-point operation and the majority of these codes employed double-precision data, the maximum rate that could be expected from the R10K processor executing a typical NAS CFD code would be 45 MFLOPS. But the Event Counter measurements show that even a high-performance, R10K workload interval requires several memory operations to produce a floating-point operation, so the measured average CPU rates in the 20-30 MFLOP range are quite reasonable. On this architecture, optimization efforts try to generate codes which do not rely as strongly as the Cray codes on memory operations to produce their floating-point operations.

6. Discussion

A comparison of workload performance requires key performance data. Table 7 contains service history data of four parallel architectures installed at NAS. For the C90 *vn*, the period represents 1993-September 31,1999. The measurement period for the IBM SP2 began in June 1996 and ended in June 1997. The *hopper* preproduction service began in March 1998 and ended in September 1999. The *steger* preproduction service began in June 1998 and ended in September 1999.

The second column is the daily average of the System Activity Reports. Since the SP2 *babbage* was not a production machine during its NAS service and the Origin machines were placed into production after the data presented in this report was collected, the user load on these three machines is expected to be less than the Cray *vn*.

Column 3 represents the floating-point performance reported by the hardware. The rate represents the number of floating-point operations divided by the elapsed time required to deliver the floating-point operations. The C90 measurements reflect daily workload monitoring[12] with the Cray Hardware Performance Monitor[14]. The IBM measurement reflects daily workload monitoring of 144 RS6000 workstations using the IBM Hardware Monitor, RS2HPM software, and NAS software[15]. The SGI measurements reflect the use of the SGI Event Counter software and hardware in conjunction with locally-modified PBS software. As discussed in section 5, the SGI results were increased by a factor of 1.6 to account for the underreporting of the multiply-add operation in the Event Counter hardware.

Column 4, the system floating-point operation efficiency, is the ratio of the delivered performance in Column 3 to the peak performance and provides a comparative view of how well the workload is able to exploit the system's floating-point capability.

Column 5 of Table 6 contains the average number of CPUs utilized by a user job. A typical time period on a parallel system will see a number of user jobs, each job executing on a number of CPUs. Ideally, the CPUs requested by the average user job should be the result of continuously monitoring the executing processes on the system and then averaging the results. Section 4 describes the PBS-accounting-file approach, which supplied the CPU averages reported in column 5.

In column 6, the average user-job GFLOPS is the product of the average user-job CPUs and GFLOPS/CPU divided by the fraction of time spent in user and system mode. The division removes the effect of the idle time which is included in the elapsed time component of system GFLOPS. Thus, for the C90,

$$\text{Average User Job GFLOPS} = (3.78 \times 1.53)/(0.87 \times 16)$$

For the SP2, a lack of users in the second half of the service period and some non floating-point intensive applications contributed to the low measured *babbage* performance. Correcting for these problems would allow *babbage* to claim an average user application performance of 0.320 GFLOPS at best.

Table 7: Workload Performance Comparison of NAS Parallel Machines

Machine	user/ system/ idle	Measured System GFLOPS	System Efficiency %	Average User Job-CPUs	Average User-Job GFLOPS
16-CPU Cray C90 *vn*	82/05/12	3.78	23.6	1.53	0.415
144-CPU IBM SP2 *babbage*	63/01/36	1.30	3.39	15.6	0.226
64-CPU SGI Origin *hopper*	53/04/42	1.78*	5.56	10.2	0.497*
256-CPU SGI Origin *steger*	48/05/46	6.18*	4.83	18.3	0.833*

*- SGI includes a factor of 1.6 to compensate for underreporting floating-point operations

Steger delivers more GFLOPS than the C90, but on typical workload codes, the C90 processors deliver

about 10 times the floating-point performance of a single *steger* processor. The C90 processors deliver about 8.5 times the floating-point performance of a single *hopper* processor.

The table reports Origin system efficiencies of 4-6%, and hardware monitoring of systems outside NAS indicates such efficiencies are currently typical of general purpose computational workloads on RISC processors. The low system efficiencies are a combination of generally low application performance, about 10% of peak, and a limited user participation, about 50% user utilization. The STREAM benchmark suggests that the R10k processor can sustain a memory data transfer rate of 45 MW/sec [13]. NAS codes, assuming no data reuse and sufficient memory hardware resources, should display a performance rate of 45 MFLOPS (9% of peak) assuming 1 FLOP per memory reference. But the hardware measurements indicate the NAS codes require more than 1 memory reference to produce a floating-point operation and that limited data reuse occurs. The memory-limited floating-point performance of the NAS workload on RISC machines is not surprising. Cray devoted a considerable amount of hardware and expertise to obtaining a proper balance between CPU and memory speed and hardware performance measurements show that memory bandwidth does not limit NAS workload performance on the C90.

Measured R10k CPU performance rates are in the 20-30 MFLOP range. On NAS workload codes, the 1998 R10K processors are more effective than the 1996 Power-2 processors. The faster clock, 250 mHZ for the R10K versus 66 mHZ for the Power-2, and the larger cache, 4 mB for the R10K versus 256 kB for the Power-2, play key roles in this improvement. The table indicates that measured workload CPU performance is only moderately coupled to peak CPU performance. The SP2 RS6000 processor, with a peak performance of 266 MFLOPS, delivered about 20 MFLOPS/CPU whereas the R10k processor, with a peak of 500 MFLOPS/CPU, delivered about 26 MFLOPS/CPU. A peak performance ratio of 1.87 translated into a delivered performance ratio of 1.3.

In spite of the low efficiencies, *steger* system performance exceeded *vn* performance by a factor of 1.6 and the measured level of system utilization indicates that *steger* can deliver greater workload performance with increased CPU utilization. Perhaps experience with the NAS C90 can indicate the level of improvement the SGI Origins could display in the production period. The C90 system displayed a performance increase of about 10% from the initial rates due to a 7% increase in CPU performance to the average value of 287 MFLOPS/CPU and a smaller increase in system utilization to the current average value of 82%. These increases occurred on top of large initial CPU and utilization rates. Thus, since the NAS SGI cluster starts from a lower base, it might experience at least a 10% improvement in the production period. However, predicting floating-point performance of workloads executing on RISC machines requires a disciplined and realistic set of assumptions. Even after the SP2 experience, many at NAS, including the author, believed that *hopper* would deliver substantially more than its measured 1.78 GFLOPS. These beliefs were based on an assumption of near 100% user CPU utilization and a benchmark-based CPU performance exceeding 40 MFLOPS. As Table 7 shows, both of these assumptions were wrong-the utilization was 50% and the CPU performance was in the 20-30 MFLOPS range.

But for a typical NAS user, the workload measurements indicate the SGI Origins increase performance by a factor of about 1.2 to 2.0 over the 1993 C90 vector architecture. The measured data support the contention that the RISC processors are ready for production. To achieve C90 production-level performance, NAS RISC users must employ parallel processing. The most recent utility for expressing parallelism, OpenMP [6], requires a limited time investment to achieve proficiency in creating parallel code.

7. Conclusion

The average SGI Origin *hopper* job, which utilizes 10 processors, performs at a rate of about 1.2 times the average user job on the 1993 Cray C90 *vn*. For the average *steger* user job, which utilizes about 18 processors, the performance factor is 2.0. The Origin performance is the first instance of NAS workload measurements which show a RISC performance advantage over the vector machines for the typical user.

The SGI machines displayed an average performance of 20-30 MFLOPS/CPU with very few measured instances of high-performance code. The performance results indicate that, due to limited data reuse, most of the NAS codes execute at rates limited by memory-bandwidth. It does not appear likely that user code will display significantly increased rates in the production period. However, the highest performing user code obtained almost 20% of peak by adapting his code to make extensive use of the SGI numerical libraries, and other users may want to employ this approach.

During the preproduction period, the Origin machines displayed a 50% CPU utilization, and this level of user activity is typical for RISC architectures used as general purpose supercomputers. Administrators ought to be able to increase this loading, both through accounting incentives, which make it easier for users to employ large numbers of processors, and through checkpointing, which makes it easier for administrators to service the larger jobs. There is a large body of NAS codes which must yet be parallelized to give C90 equivalent performance, and it may be useful for administrators to provide a weekly informal seminar to assist in the conversion of these codes.

8. References

1. R.J. Bergeron, "The Performance of the NAS HSPs in 2Q93", http://science.nas.nasa.gov/Pubs/TechReports/RNDreports/RND-94-005/RND-94-005.html
2. NAS/ACSF Scientific Consulting Group, "Portable Batch System (PBS)", http://parallel.nas.nasa.gov/Parallel/PBS/.
3. SGI,"Performance Tuning Optimization for Origin2000 and Onyx2",http://techpubs.sgi.com/library/manuals/3000/007-3511-001/html/O2000Tuning.0.html.
4. J. Laudon and D. Lenoski, "System Overview of the SGI Origin Product Line", http://www.sgi.com/origin/whitepapers.html,http://www-unix.mcs.anl.gov/mpi/.
5. The MPI Forum, "The Message Passing Interface (MPI) Standard", http://www-unix.mcs.anl.gov/mpi/.
6. The OpenMP Architecture Review Board, "The OpenMP Standard", http://www.openmp.org/.
7. B. Traversat, private communication[1998].
8. R. Jain, "The Art of Computer Systems Performance Analysis", Wiley Publishing, 1991.
9. M. Zagha, et al, "Performance Analysis Using the MIPS R10000 Performance Counters, "Proceedings: Supercomputing'96, Pittsburgh, PA, (November 1996).
10. M.P. Anant, MRJ, private communication, December 1999.
11. J. Taft,"Initial SGI Origin2000 Tests Show Promise for CFD Codes", http://www.nas.nasa.gov/Pubs/NASnews/97/07/article01.html.
12. W. Wulf and S.A. McKee, "Hitting the Memory Wall: Implications of the Obvious", Computer Architecture News, 23(1):20-24, March 1995.
13. J. D. McCalpin, "STREAM: Sustainable Memory Bandwidth in High Performance Computers", http://www.cs.virginia.edu/stream/
14. Cray Research, Inc. UNICOS Performance Utilities Reference Manual, SR-2040 7.0, 1992.
15. R.J. Bergeron, "Measurement of a Scientific Workload Using the IBM Performance Monitor", SC98, Orlando Fla., Nov. 8-13, 1998. http://www.supercomp.org/sc98/TechPapers/sc98_FullAbstracts/Bergeron1289/index.htm

Figure 1

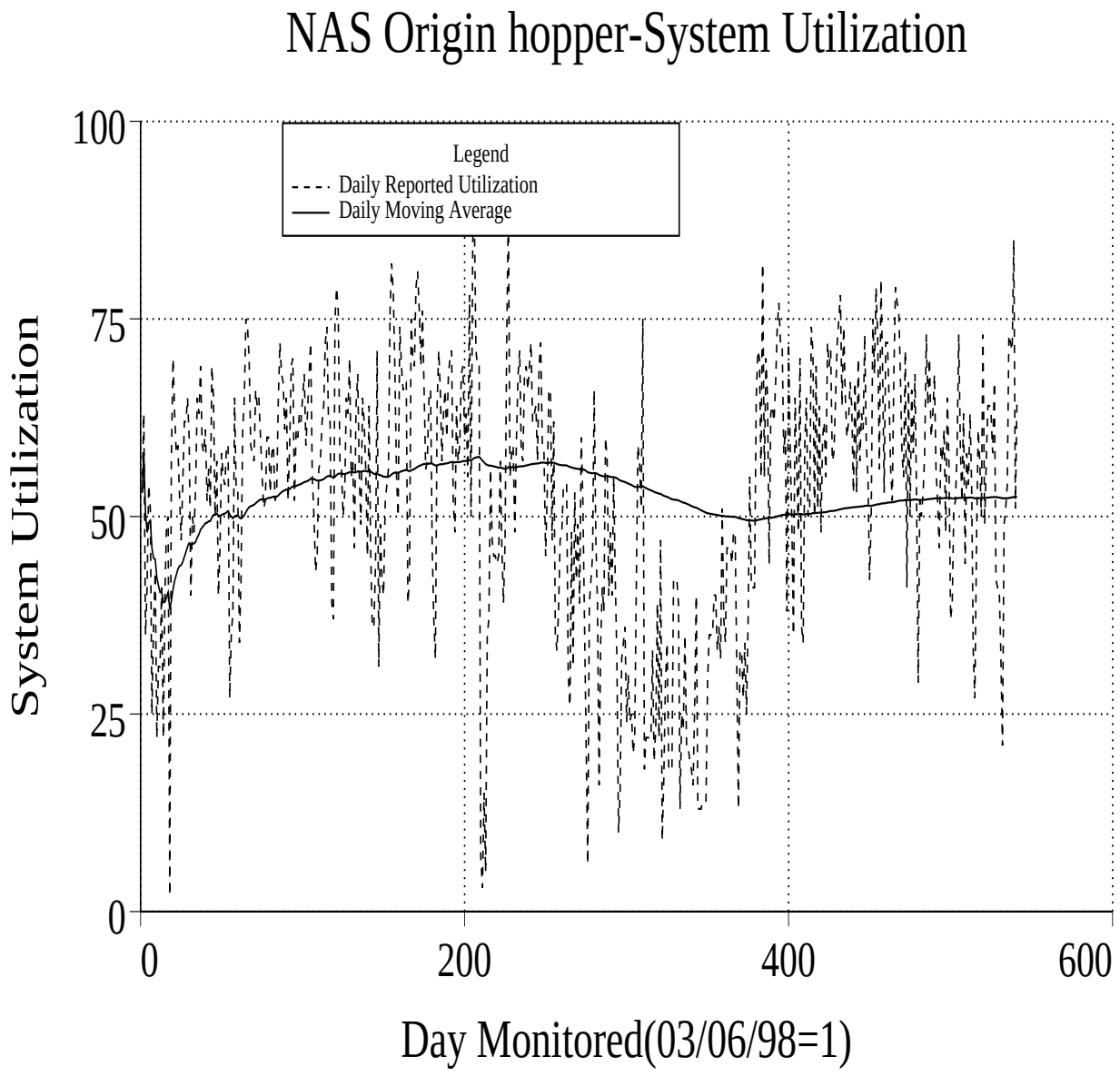

NAS Origin hopper-System Utilization
Legend
Daily Reported Utilization
Daily Moving Average
System Utilization
100
75
50
25
0
0
200
400
600
Day Monitored(03/06/98=1)

Figure 2

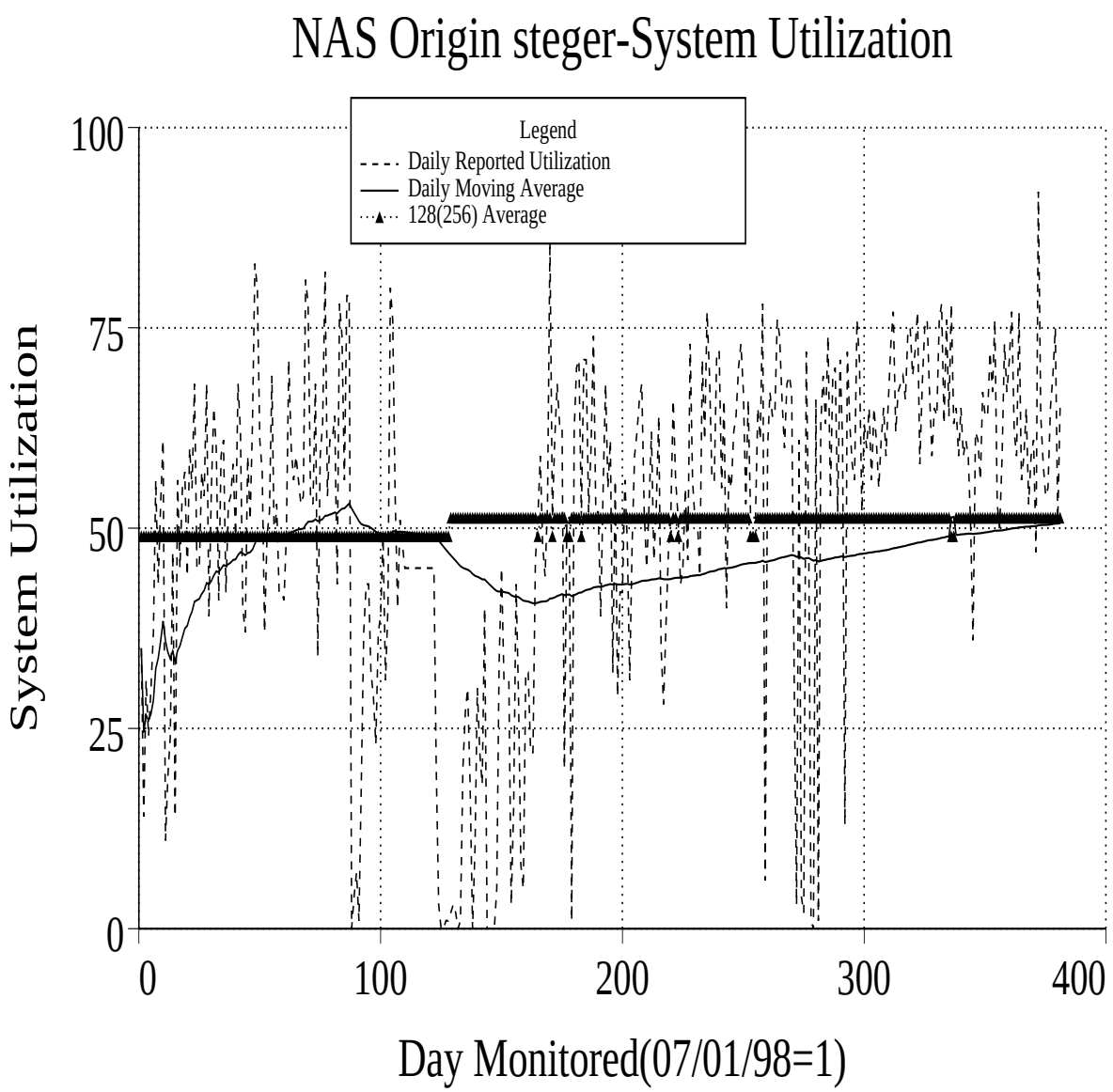

NAS Origin steger-System Utilization
Legend
Daily Reported Utilization
Daily Moving Average
128(256) Average
System Utilization
100
75
50
25
0
0
100
200
300
400
Day Monitored(07/01/98=1)

Figure 3

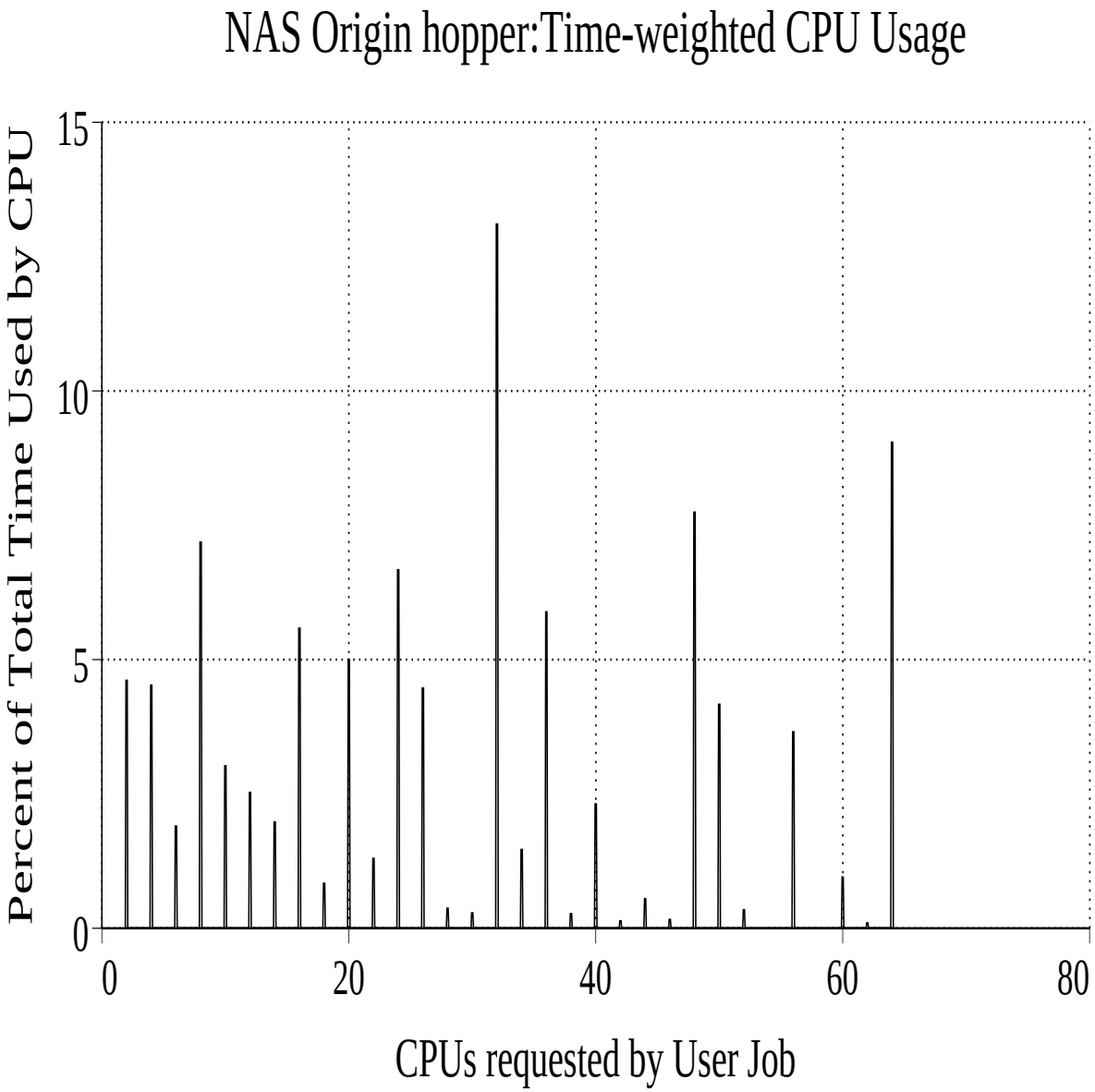

NAS Origin hopper:Time-weighted CPU Usage
Percent of Total Time Used by CPU
15
10
5
0
0
20
40
60
80
CPUs requested by User Job

Figure 4

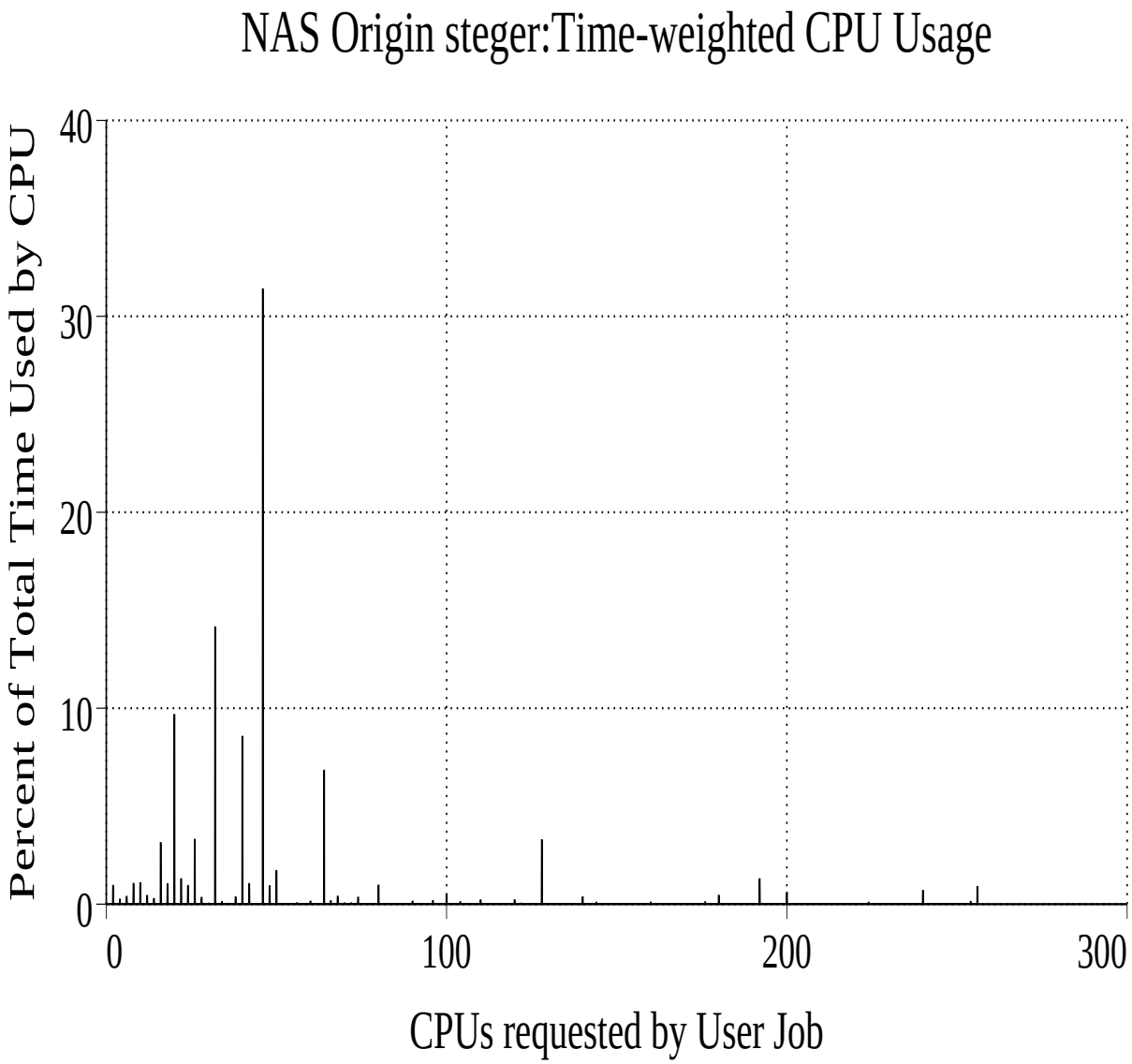

NAS Origin steger:Time-weighted CPU Usage
Percent of Total Time Used by CPU
40
30
20
10
0
0
100
200
300
CPUs requested by User Job

Figure 5

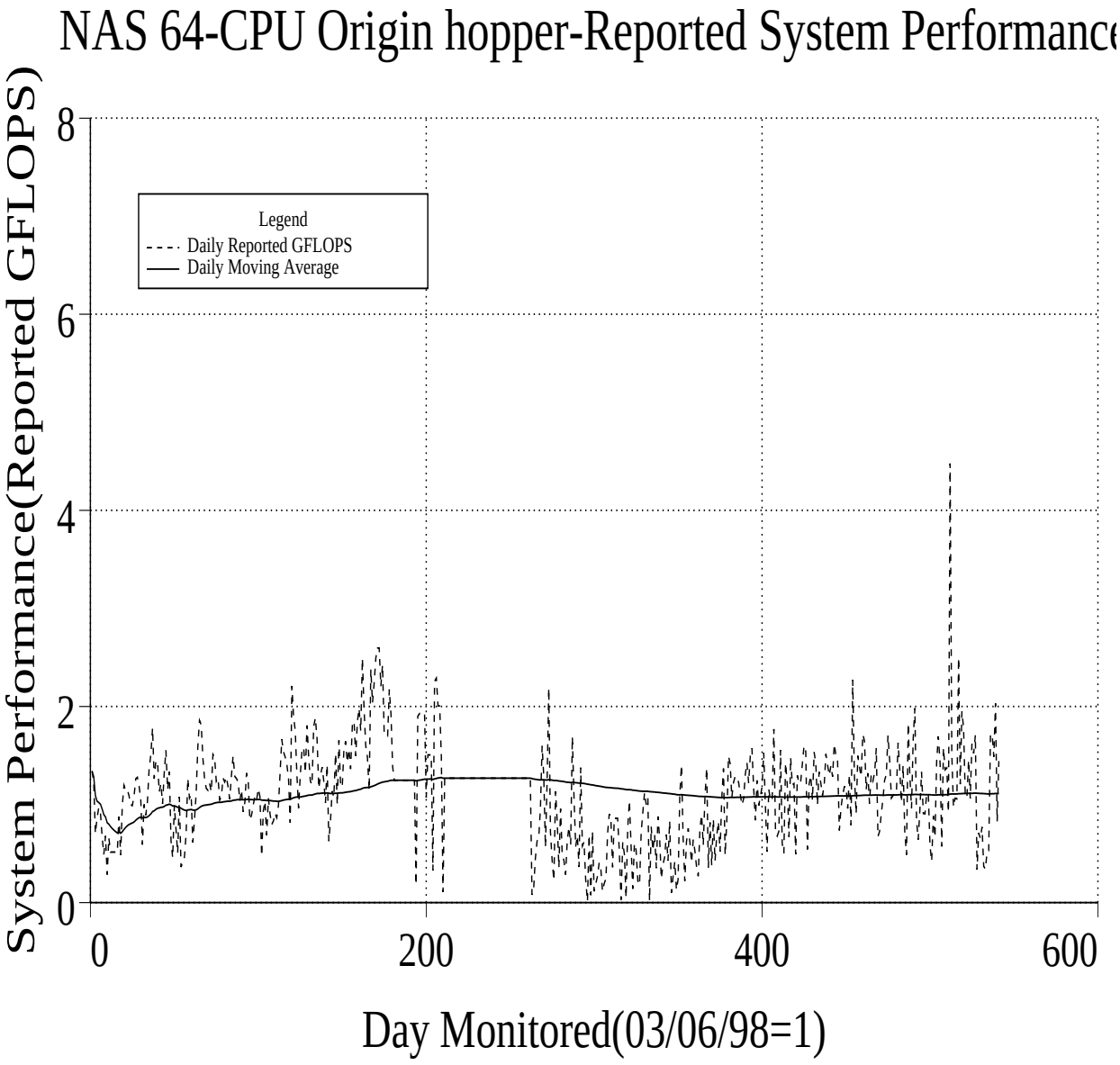
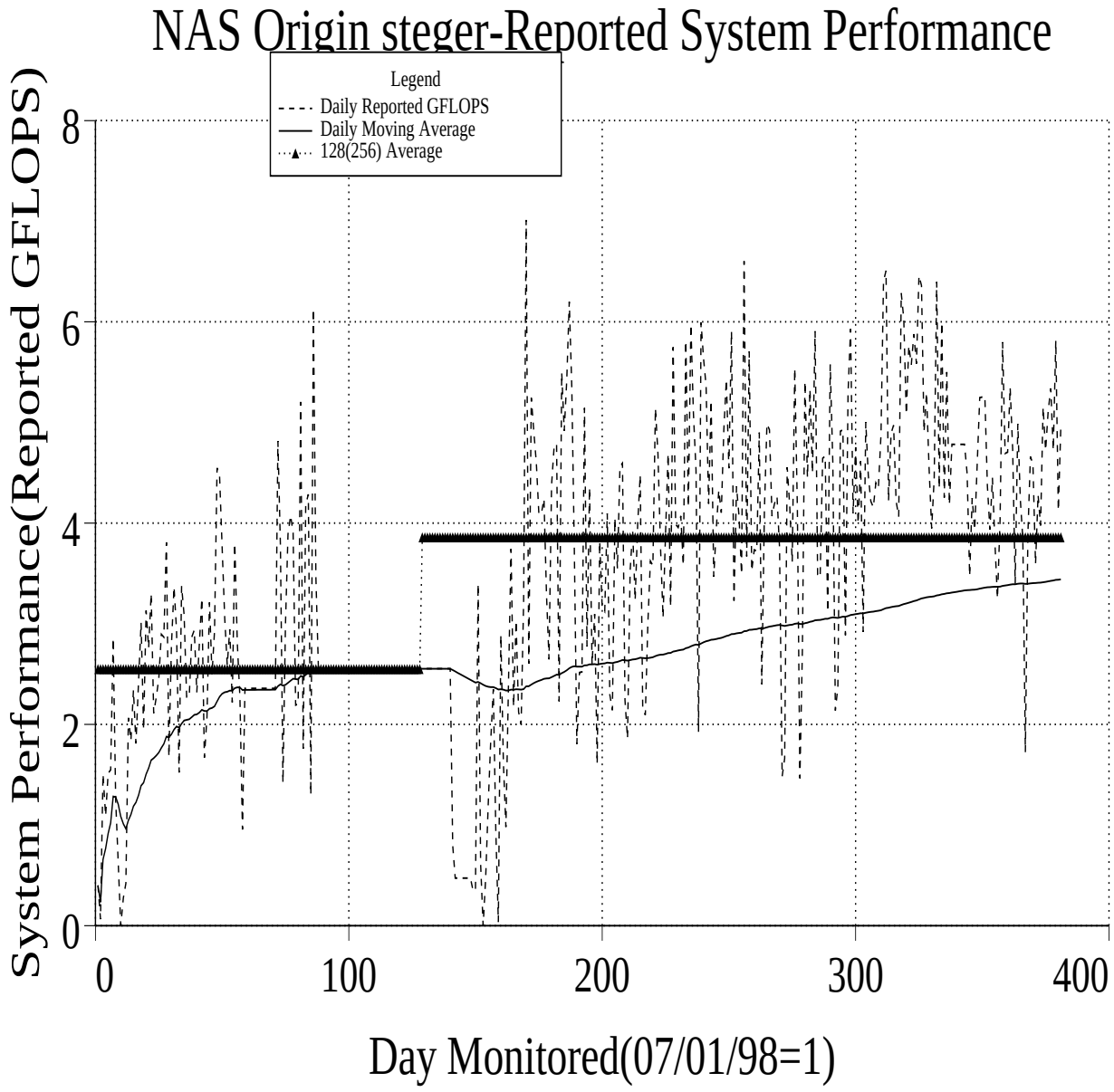

NAS 64-CPU Origin hopper-Reported System Performance
System Performance(Reported GFLOPS)
Legend
Daily Reported GFLOPS
Daily Moving Average
0
2
4
6
8
0
200
400
600
Day Monitored(03/06/98=1)

Figure 6


NAS Origin steger-Reported System Performance
Legend
Daily Reported GFLOPS
Daily Moving Average
128(256) Average
System Performance(Reported GFLOPS)
0
2
4
6
8
0
100
200
300
400
Day Monitored(07/01/98=1)